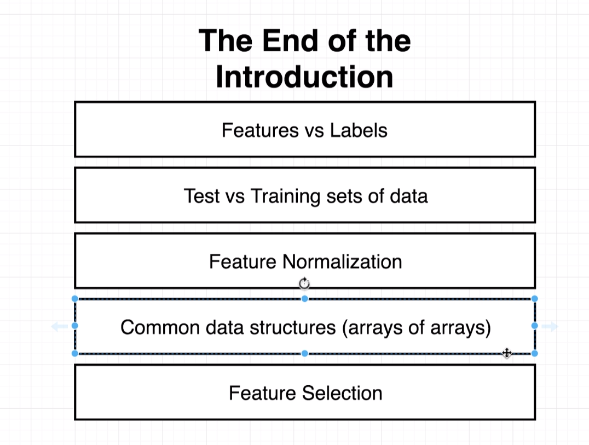


**What we will do over and over:**

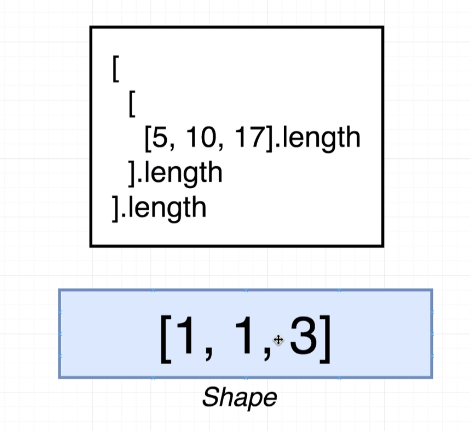
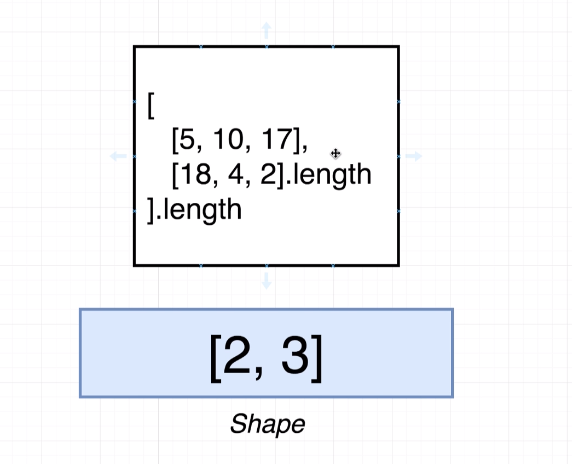
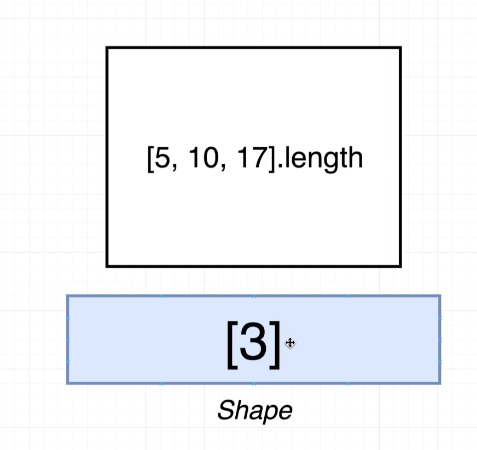


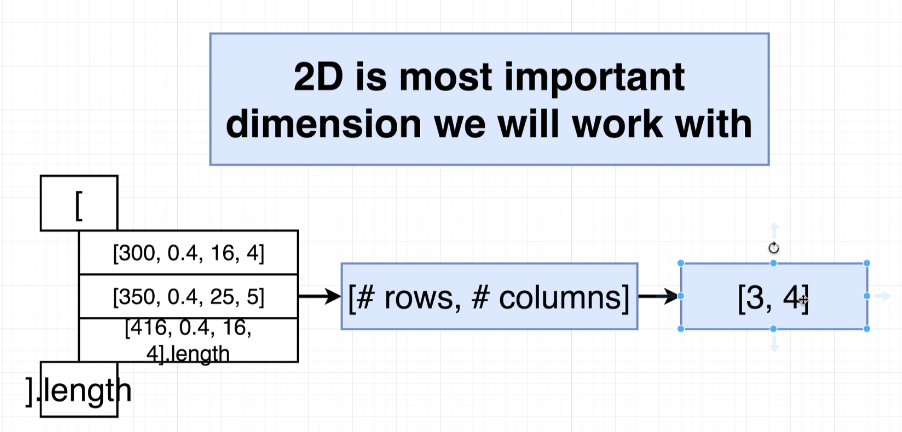
**Tensor:** an object that wraps a group of numbers. Inside of a tensor you can have a deeply nested array, a flat array and an array inside of an array.

**Dimensions:** The number of dimensions a tensor has is based on how many “levels” of opened array brackets a tensor has.

**Shape:** How many records in each dimension. Imagine calling “.length” one time on each dimension from the outside of a tensor.

**Ex.**





Data.get(row, column); // retrieves that element from the matrix.

Data.slice([start column, ending column], [number of rows, elements wide the slice has])

// this returns the column desired.

Ex.

Data.slice([0,1], [6,1]) ;

Data.shape; // returns the number of rows and columns you have in this format [rows, columns]

The value **-1:** means give me as many rows/columns that you can.

**Join together two tensors:**

tensorA.concat(tensorB, 0); // stacks the arrays

tensorA.concat(tensorB, 1); // puts them together length-wise

**Summing up values in a tensor:**

Data.sum(0); // This will sum the columns and output a tensor with those summations.

Data.sum(1); // This will sum the rows and output a tensor with the row summations.

**\*\***When summing a tensor, it will reduce the dimension to a 1D from a 2D**\*\*** **UNLESS!** You use the second optional argument of “true”. This will make the tensor keep its original dimension after summation.

**Increase tensor Dimension:**

TensorData.expandDims(); // this will increase dimension by 1.

And… if you insert a “1” as an argument. It will switch the rows to columns.

**Accessing data within the built tensor:**

console.log(data.print());

**unstack()** method will put the tensor into a regular array and will be treated as a regular array.

**KNN Algorithm Steps:**

1. Find distance between features and prediction point.
2. Sort from lowest point to greatest
3. Take the top K records
4. Average the label value of those top K records.

**Standardization** is going to solve our extreme square foot differences.

**Standardization :** (value – Average)/(standardDeviation)

**Standard deviation** is the square root of variance.

**Linear Regression and Gradient Descent:**

**Mean squared error equation:**

Summation of all values of ((Guess – Actual)^2) / number of guesses

**\*\* The better the guess is, the closer the MSE will be to zero\*\***

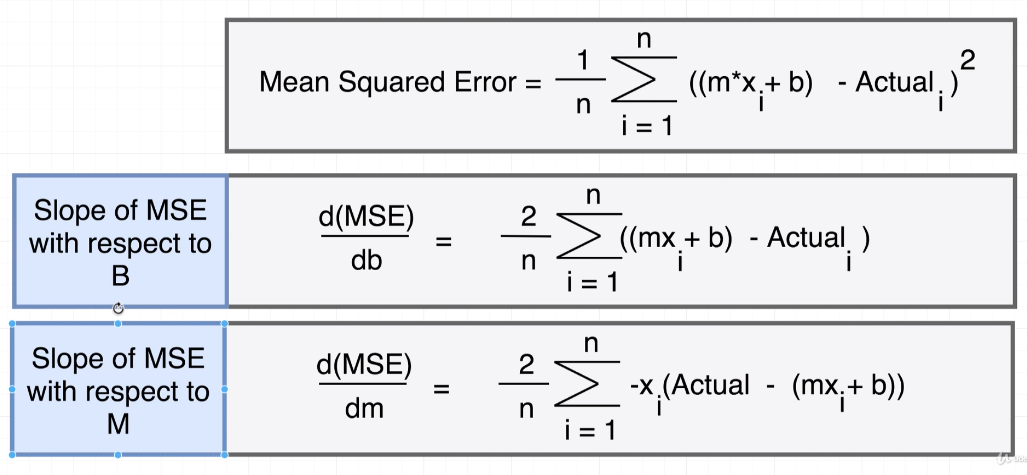
Ex. output = m \* Feature + b

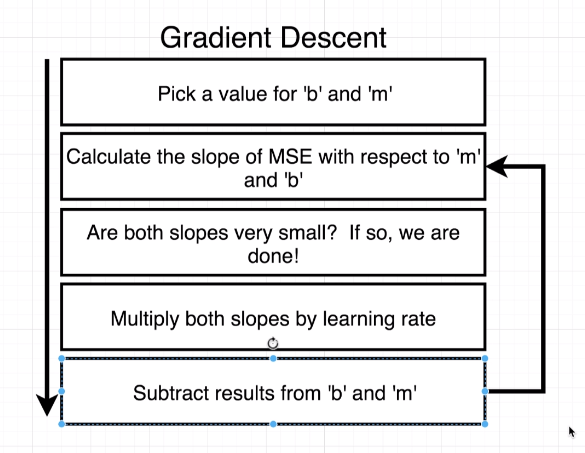
Using derivatives and MSE and assuming our m is 0.

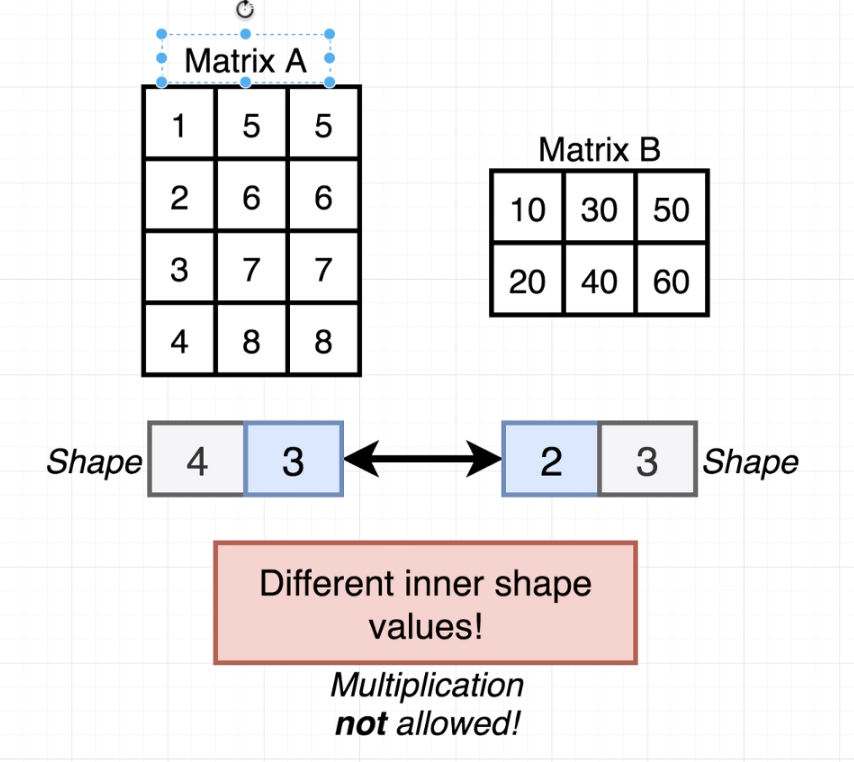
(2/n) \* (b – Actual)

**Gradient Descent:**

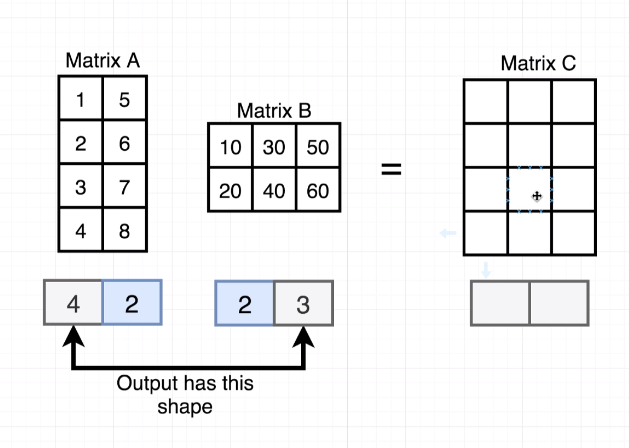
**Excel Example:** <https://docs.google.com/spreadsheets/d/19y-y19yuPXo2VpOVpnKVOi7ojmdeJ-ObCokzZCLF6XA/edit#gid=0>



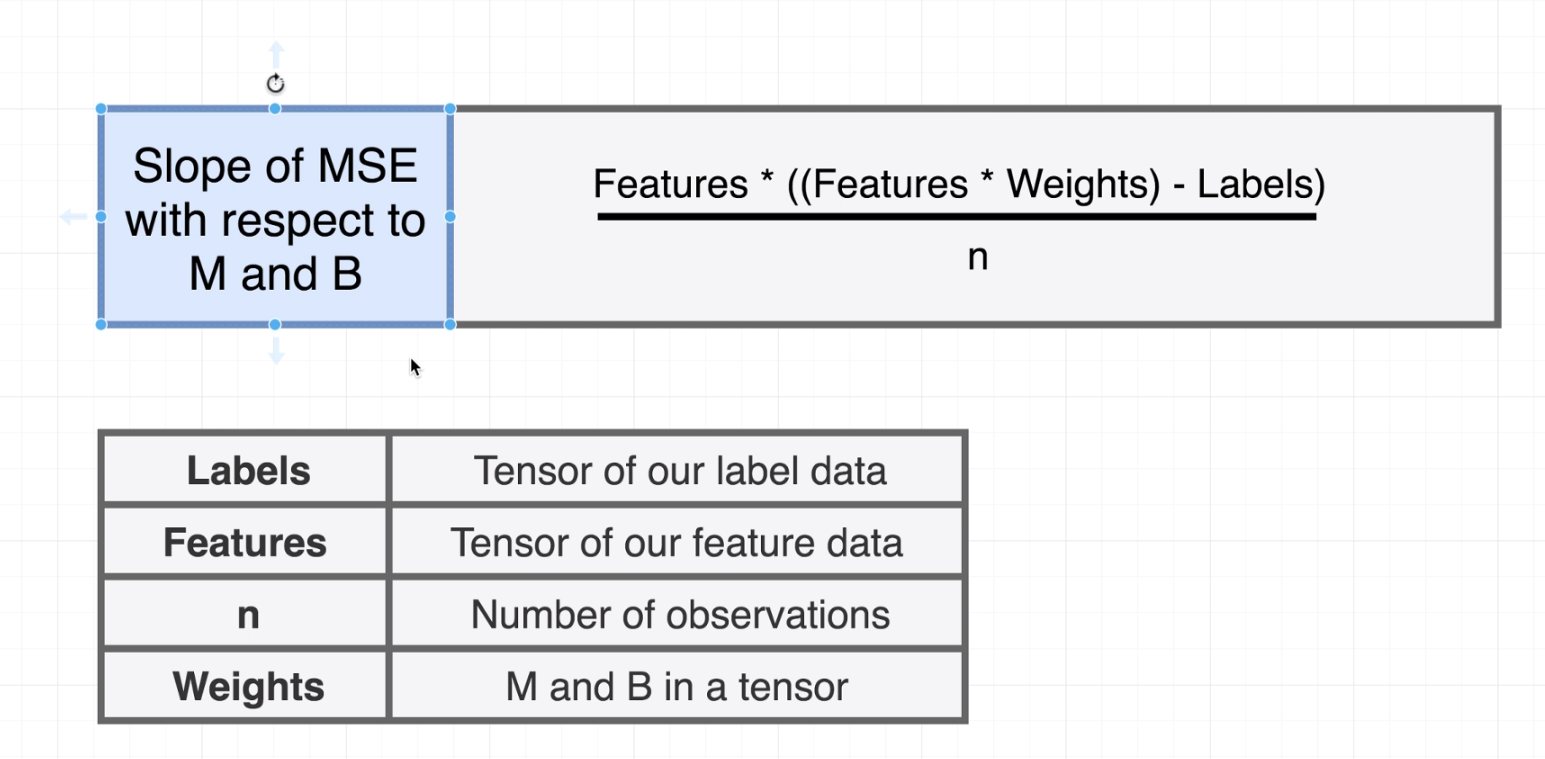


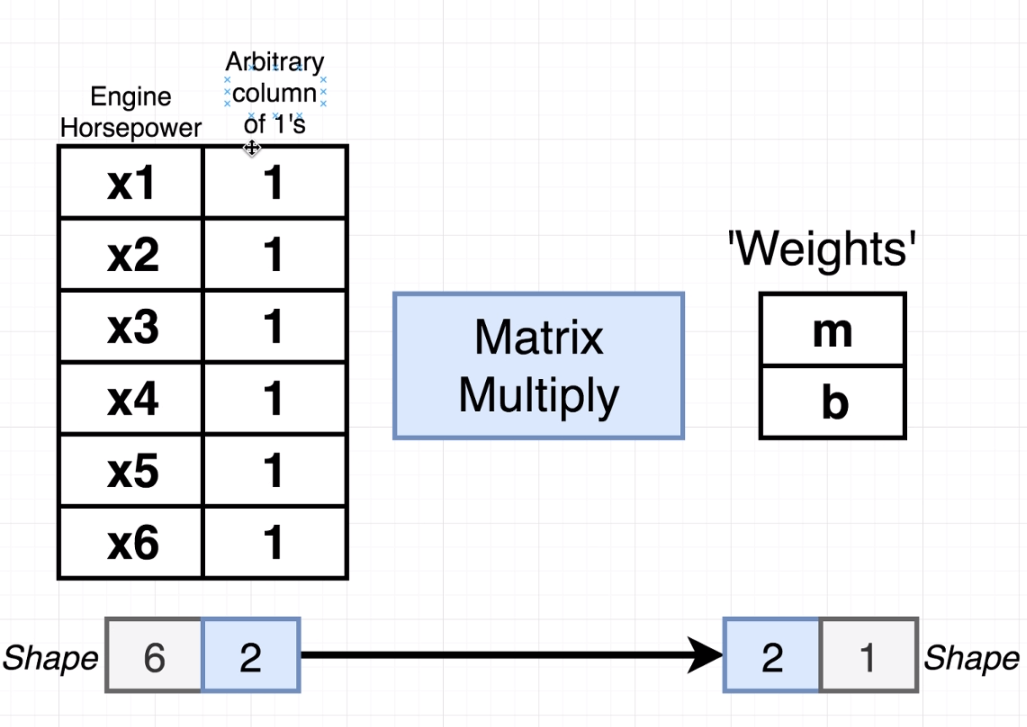


The numbers next to each other in the top example, numbers 2 and 3 are different so those two matrices cannot be multiplied together.

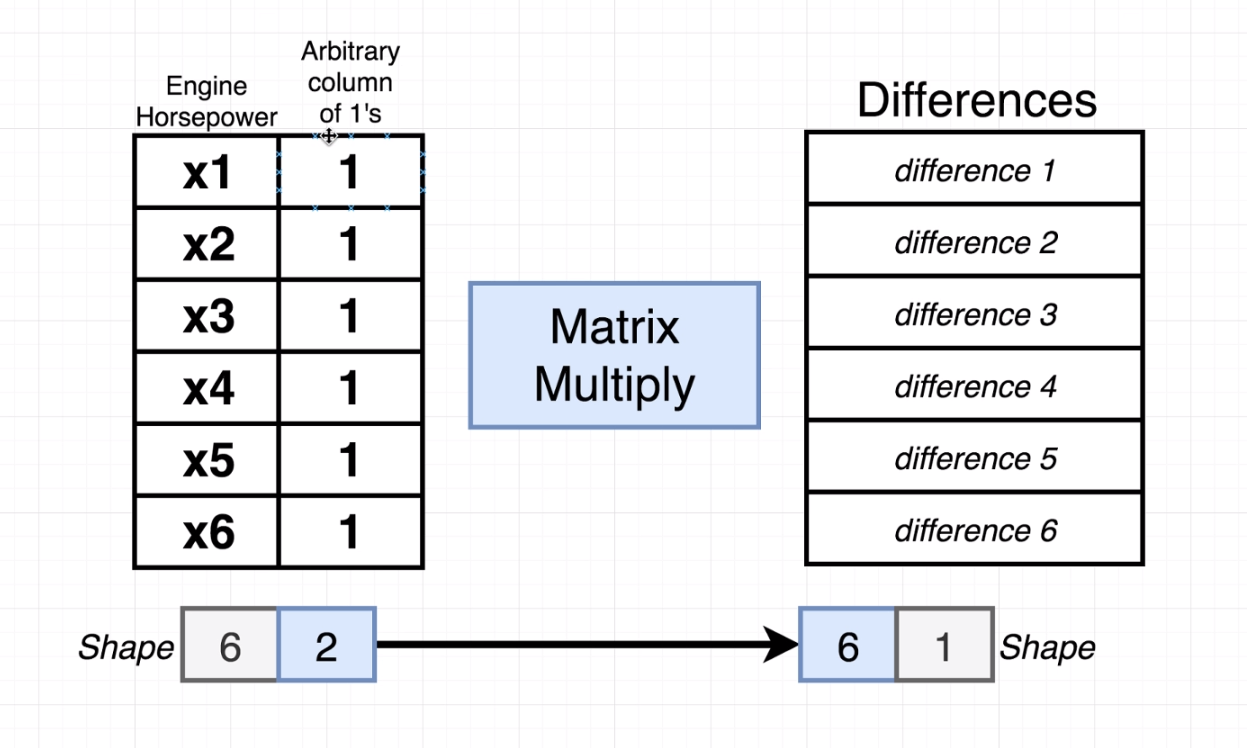


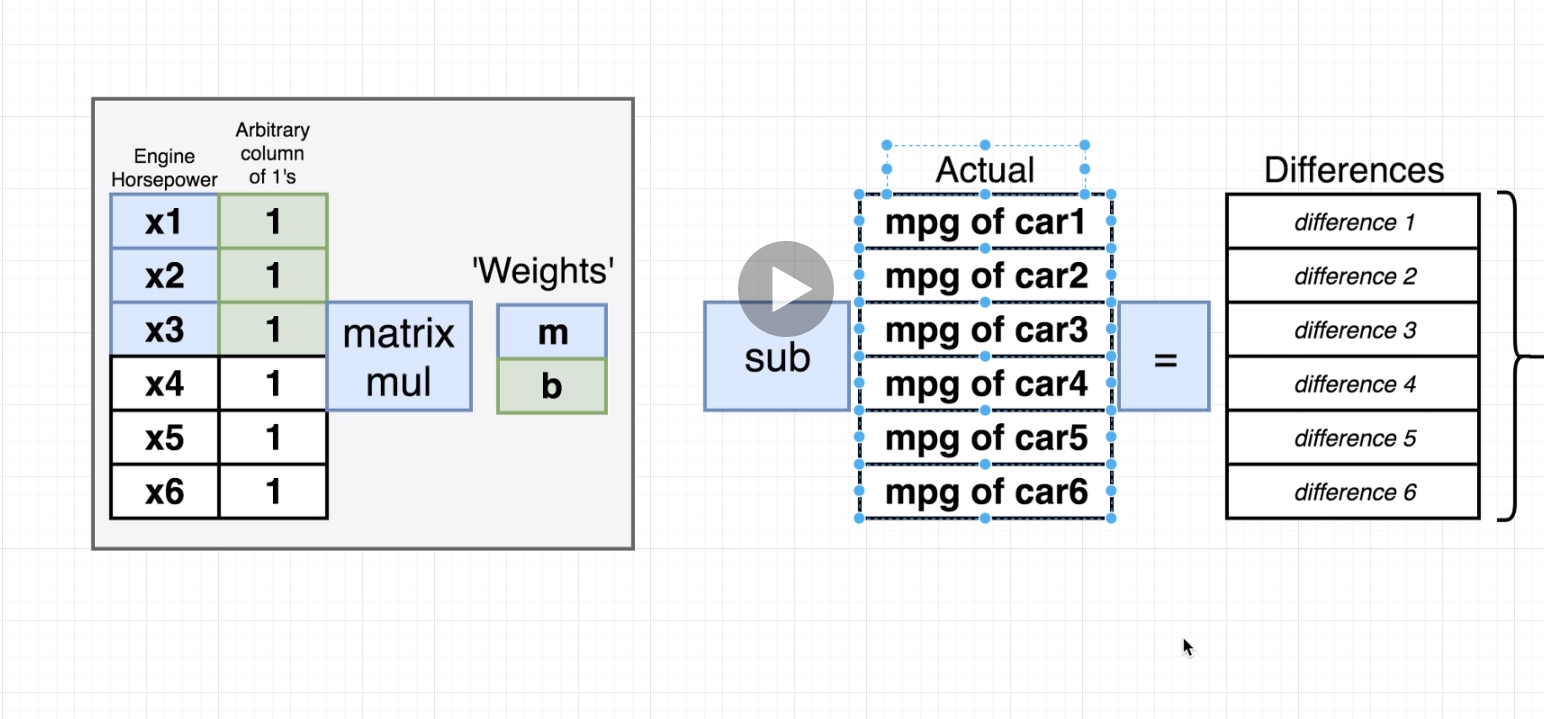
This example works. And the resulting shape of the matrix will be the row of the first matrix by the columns of the second matrix. [4,3]



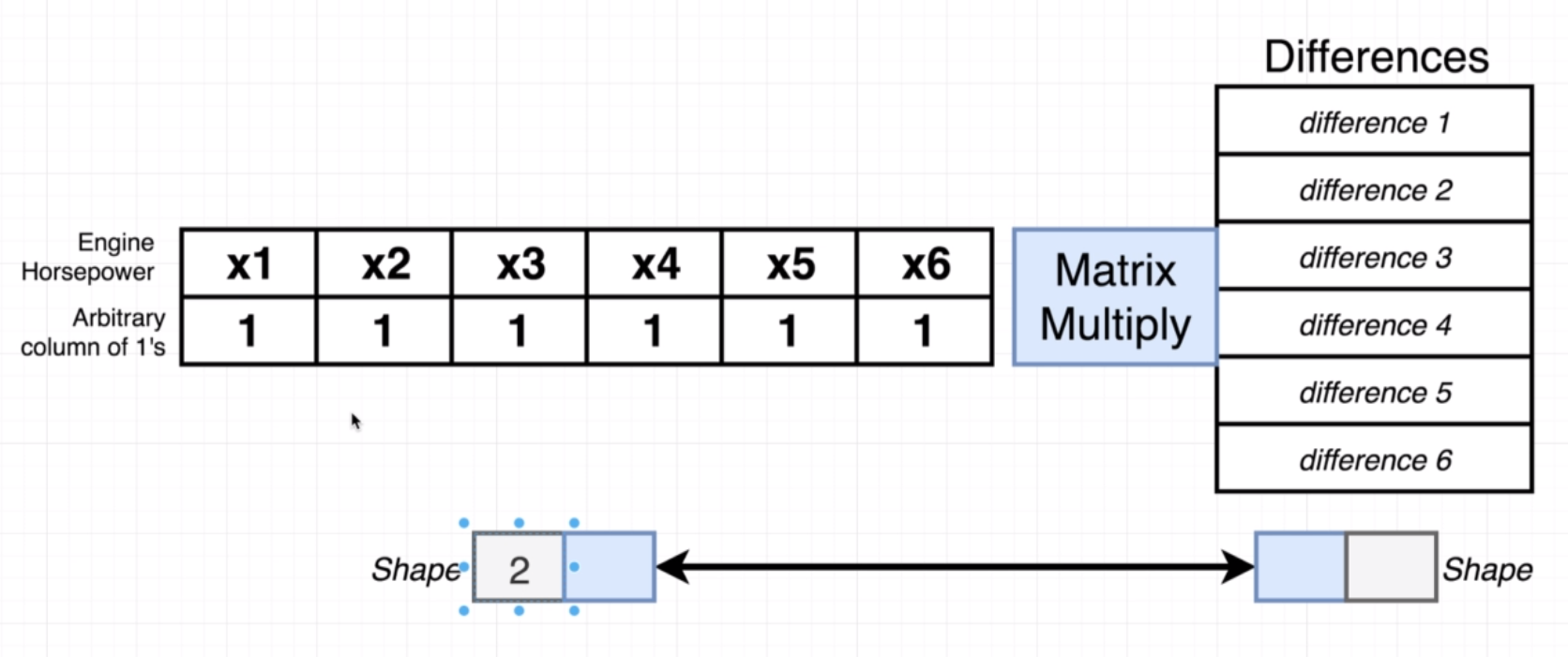


Adding this column of ones lets us multiple the feature by our weights giving us the **mx + b**



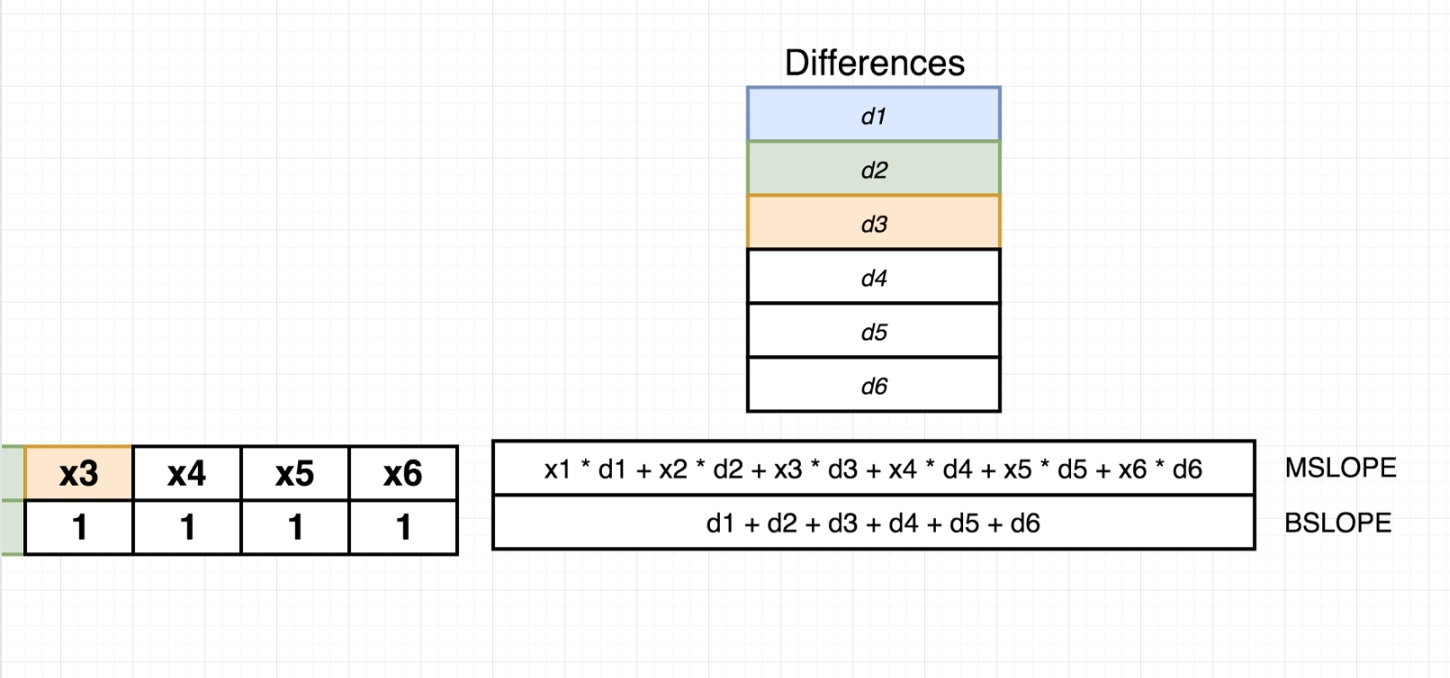


**Transposing matrices**



Transposing the matrix on the left will turn its rows into columns and columns into rows.

Doing this, we are able to calculate our MSLOPE and BSLOPE together. In the final matrix.



To generate a column of 1s: tf.ones([])

**Questions:**

**App Ideas:**

* **Future me:** using machine learning it will show you how you will look a certain amount of years from now. Detailing health problems to look out for.
  + If implementing a specific workout regiment, how would that change your look.
  + If going out in the sun, x hours per day.
  + Specific Diet
  + Providing historical pictures.